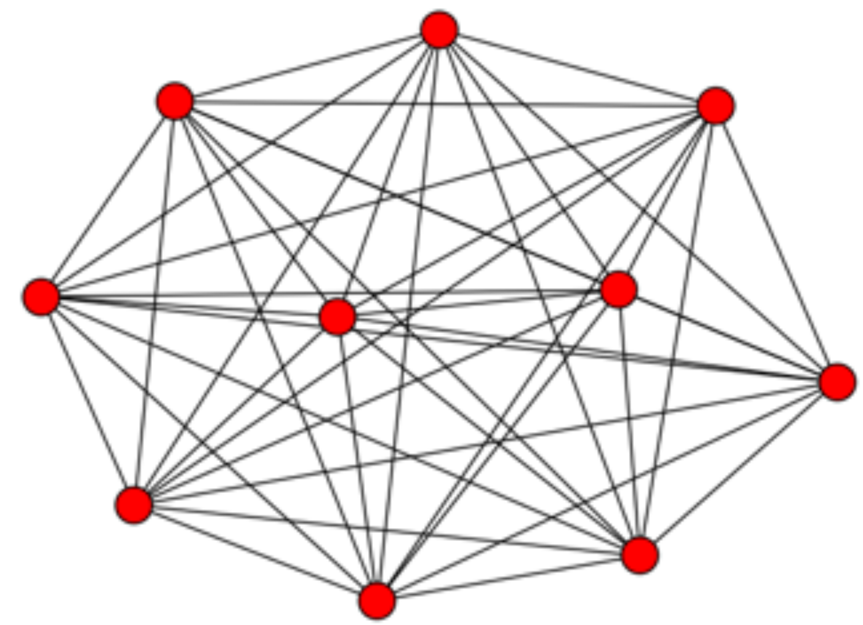
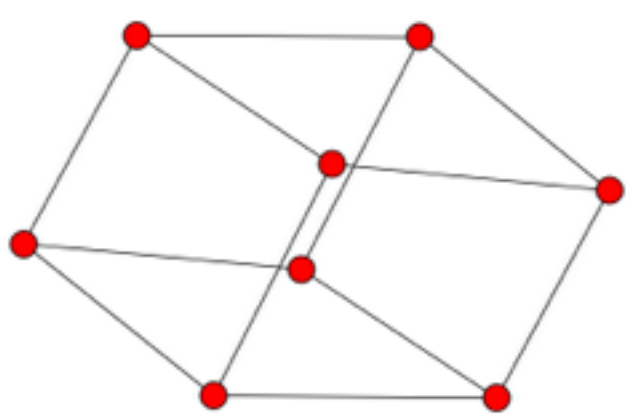
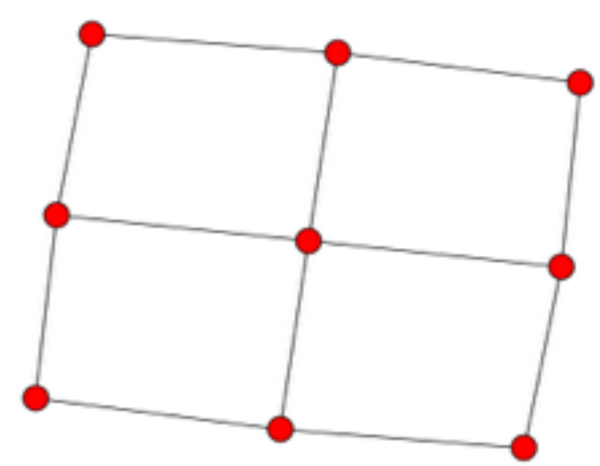
Regular Symmetry Patterns

Anthony W. Lin (Yale-NUS), Khanh Nguyen (Autocad)
Philipp Ruemmer (Uppsala), Jun Sun (SUTD)

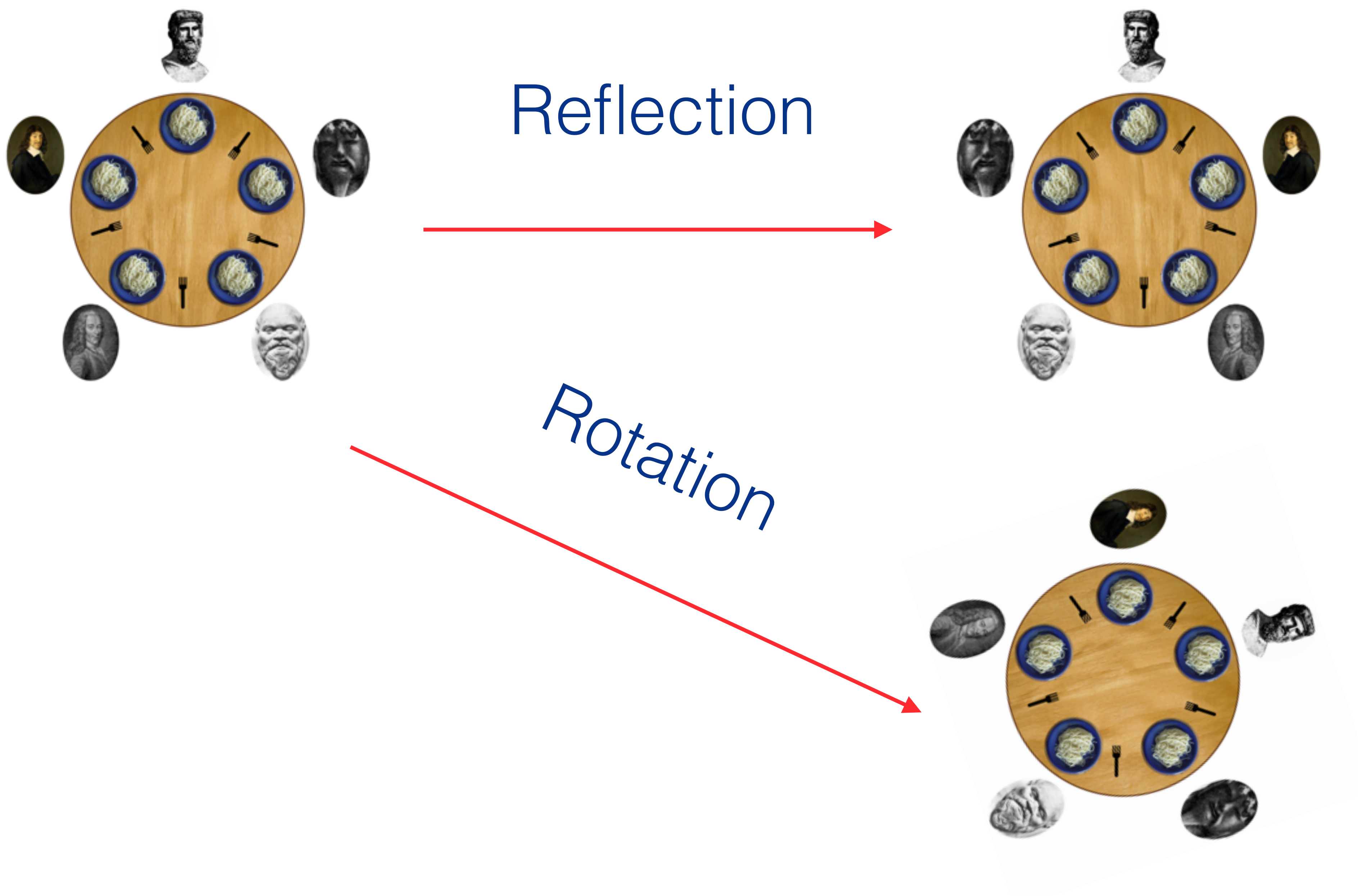
Symmetries in systems







Symmetry examples



Symmetries are closed under composition

Symmetries as automorphisms

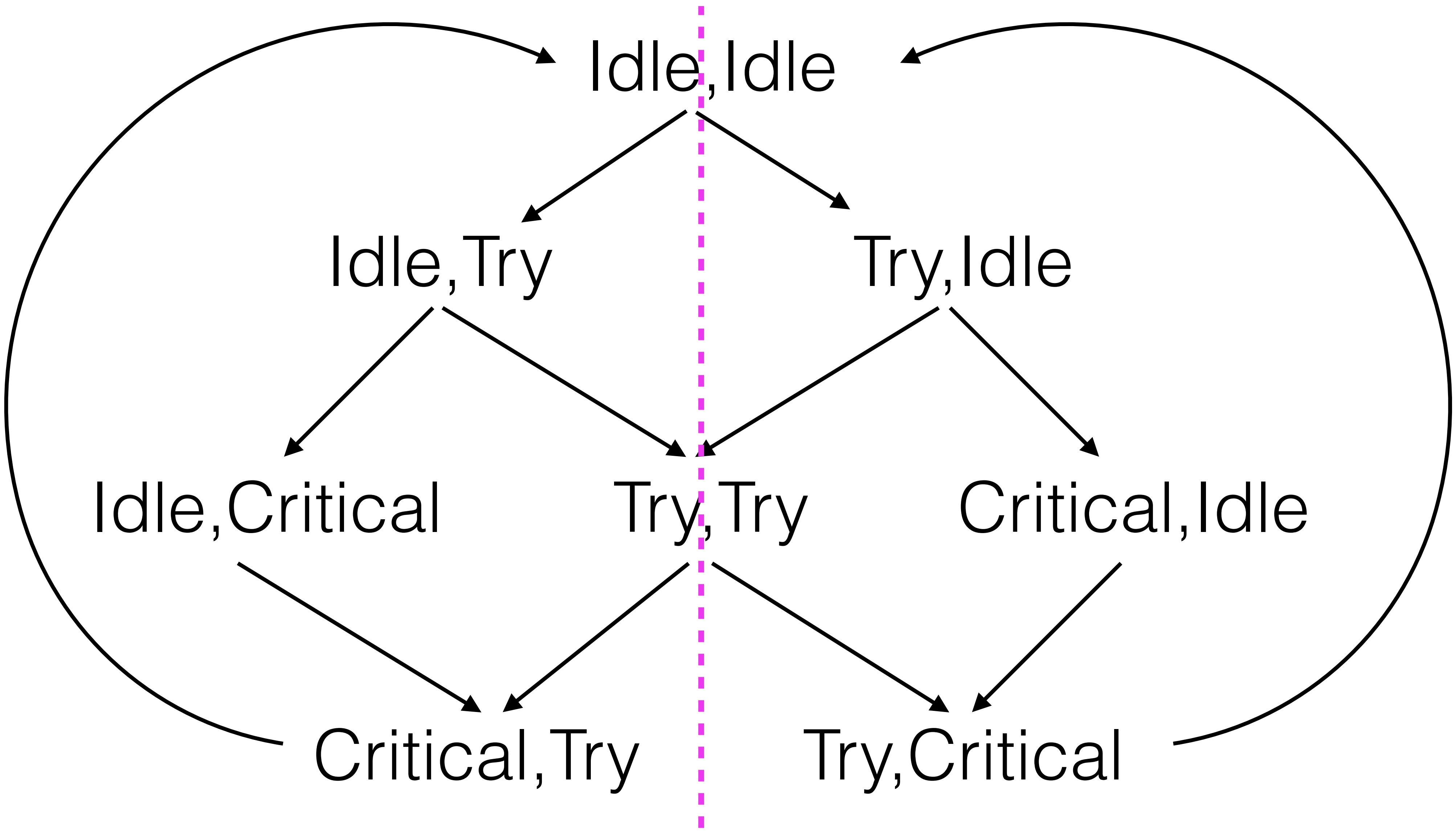
Automorphism: structure-preserving bijection on system configurations by permuting indices

$$\pi : 1 \mapsto 2 \mapsto 3 \mapsto \cdots \mapsto n \mapsto 1$$

(Critical)(Idle)(Idle) ——> (Idle)(Critical)(Idle)

The behaviour of systems is *indistinguishable* under an automorphism

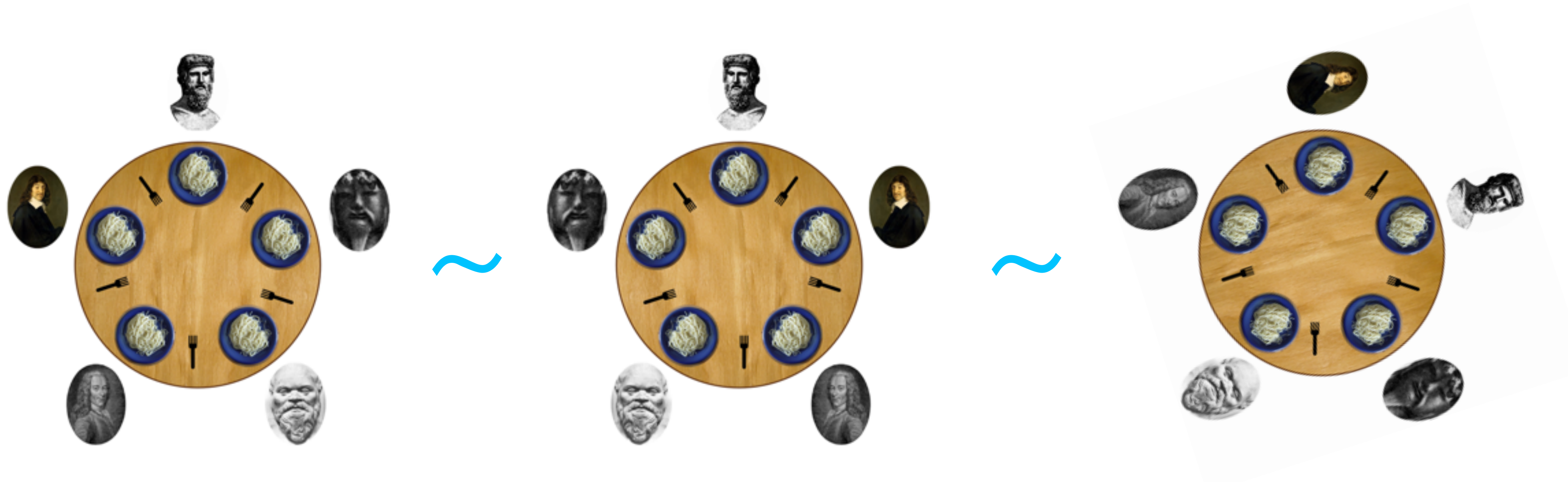
Automorphism example



Symmetry: 1 —> 2 —> 1

Symmetries help model checking

Gist: Prune branches from states in the same equivalence class as visited states



The space reduction can be exponential!

Works on all properties (safety, liveness, …)

Two problems

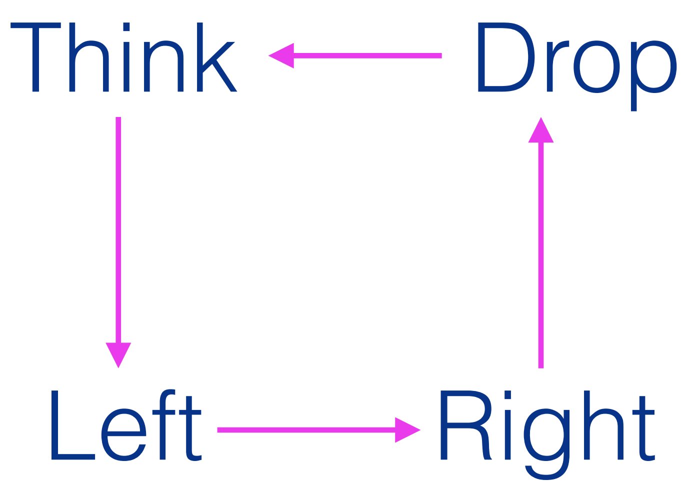
Say, we mainly attack the first problem and, to some extent, the second problem.

- **Symmetry identification**: how to identify symmetries in a given system
- **Symmetry exploitation**: (1) once symmetries are identified, check two states are similar (up to symmetries), (2) compute the "quotient" systems

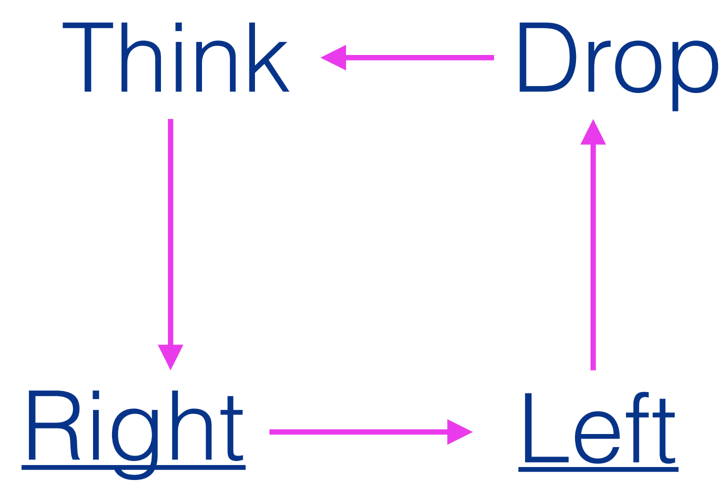
Both problems are in general computationally difficult!

Challenge: devise practical solutions to the problems

Concurrency by Replication



Template 1



Template 2

Parameterised systems

Instance with *any* number of processes can be obtained by replicating templates (a.k.a. parameterised systems)

Definition: an *infinite* family of finite-state systems

....

Parameterised Systems Help Verification

<u>Instance-by-instance (using finite-state model checkers)</u>:

Size 1	0.1s
Size 2	0.1s
…	
Size 5	1.5s
…	
Size 10	62s
…	
Size 15	Timeout

<u>Parameterised verification (regular model checking, etc.)</u>:
Replication tends to produce "similar correctness proofs" for each size and can be symbolically represented

Success on safety, but not so on other properties (e.g. liveness)

Can Parameterised Systems Help for Symmetry Finding?

Instance-by-instance (using finite-state symmetry finders):

Size 1	0.01s
Size 2	0.01s
…	
Size 5	0.2s
…	
Size 15	80s
…	
Size 20	Timeout

Parameterised:
??

Symmetry “Patterns” for Parameterised Systems

Observation:

Instances of parameterised systems (obtained by) replications tend to exhibit *similar-looking symmetries*

Pattern Example: Rotation

These 5 symmetries (case n=5) can be generated by

$$\sigma_5 : 1 \mapsto 2 \mapsto 3 \mapsto 4 \mapsto 5 \mapsto 1$$

For general n, this rotation symmetry pattern is

$$\sigma_n : 1 \mapsto 2 \mapsto 3 \mapsto \cdots \mapsto n \mapsto 1$$

Pattern Example: Reflection

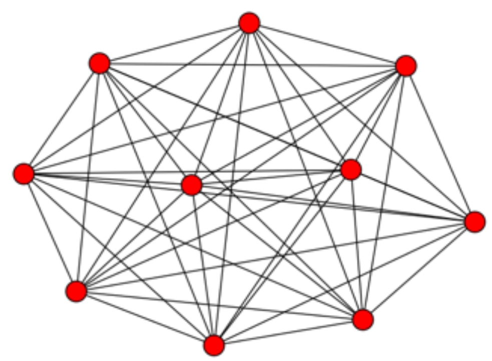
These 2 symmetries (case n=5) can be generated by

$$\pi_5 : (1,5)(2,4)(3)$$ (in cycle notation)

For general n, the reflection pattern is

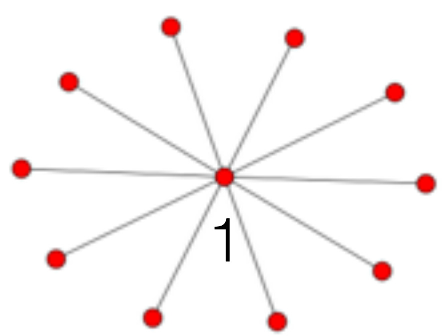
$$\pi_n : (1,n)(2,n-1)\cdots(\lfloor n/2 \rfloor, \lceil n/2 \rceil)$$

Other patterns



Broadcast protocol

Full symmetry (all permutations on {1,…,n})



Resource allocator

Full symmetry on subsystem (all permutations on {1,…,n} that fix the center point 1)

Contributions

Symbolic Framework for Symmetry Patterns in Parameterised Systems

<u>Language for Describing Systems</u>: letter-to-letter transducers (standard in regular model checking)

<u>Language for Describing Symmetries</u>: letter-to-letter transducers (NEW)

Expressive for describing practical symmetry patterns

automatic verification and synthesis of symmetry patterns

Symmetry verification

Does the given parameterised system exhibit …?

- Rotations
- Reflections
- Full symmetries
- Above symmetries in a subsystem …

Key Contribution: *Each can be expressed and automatically checked in our framework!*

Good news: there is a “library” of common symmetries

Symmetry synthesis

Symmetries in parameterised systems may not be obvious …

- Data symmetries (e.g. fork position swapped)

- Symmetries in a subsystem (but which?)

Contribution: *a CEGAR method for synthesising symmetry patterns in a parameterised system*

The symbolic framework: more technical details

Transducers

(Finite) Automata over the alphabet $\Sigma \times \Sigma$

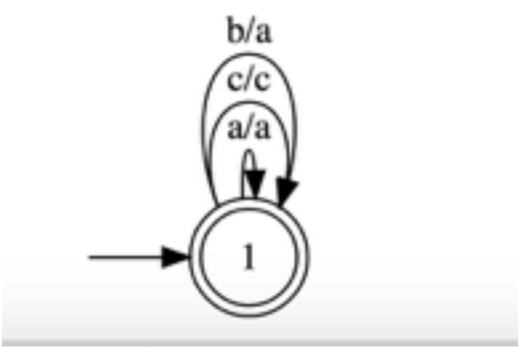
Symbolic representations of infinite binary relations

Example:

$$\Sigma = \{a, b, c\}$$

$$R = \{(v, w) : w \text{ is } v \text{ with } b \text{ replaced by } a\}$$

Automaton:



a b c

a a c

Automatic transition systems (Regular Model Checking)

Set of states: Σ^* (or a regular subset thereof)

Labelled transitions: defined by a finite family of transducers (one transducer for each action label)

Example: Dining-Philosopher (pick left first)

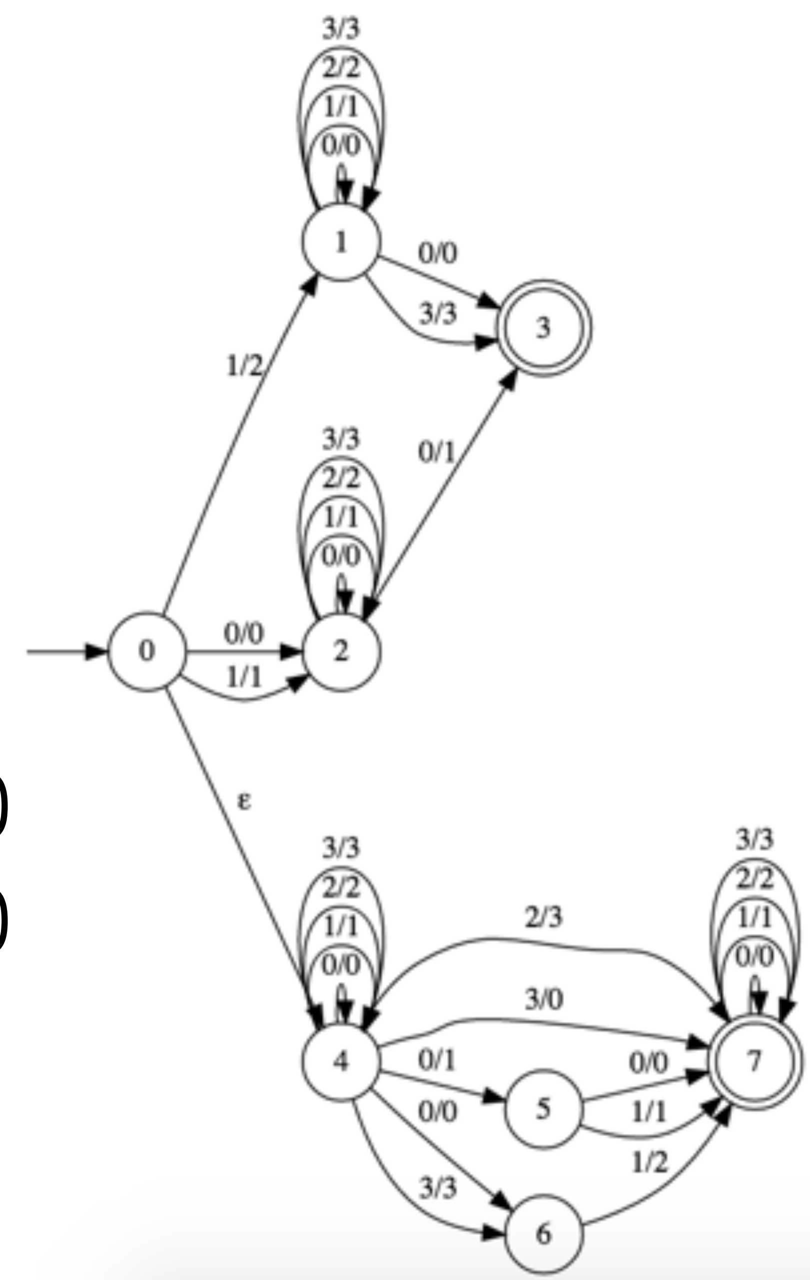
$$\Sigma = \{0, 1, 2, 3\}$$

0 - Thinking ← 3 - Drop Left

↓ ↑

1 - Pick Left → 2 - Pick Right

0	0	0
0	1	0



Symmetry Pattern

Defn: a length-preserving automorphism on an automatic transition system

$$f : \Sigma^* \rightarrow \Sigma^*$$

$$\mathrm{len}(f(v)) = \mathrm{len}(v)$$

Bijection, Homomorphism, …

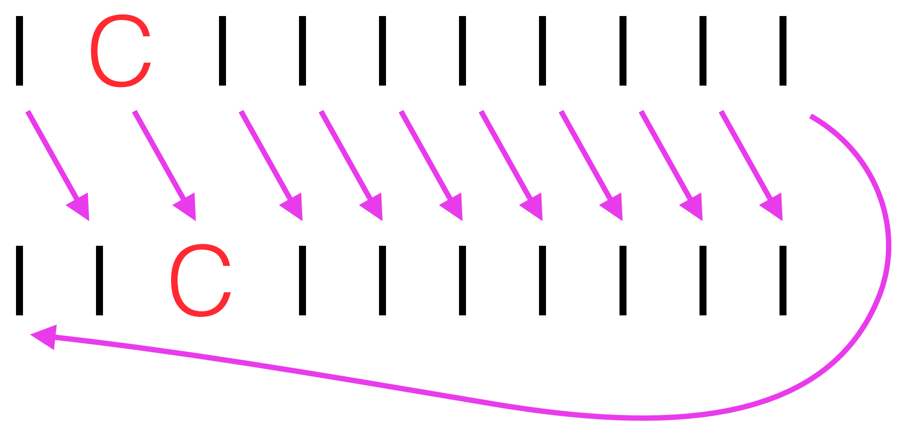
Regular Symmetry Pattern

Defn: Symmetry pattern that can be represented by a transducer

View a function as a binary relation

Examples (next few slides): rotation, swap, …

Rotation is regular



Automaton remembers when reading *i*th position:

1. *i*th position, 1st letter
2. 1st position, 2nd letter

Symmetry Pattern Verification

Verifying Regular Symmetry Patterns

Theorem: Checking whether a given automatic system exhibits a given regular symmetry pattern is PTIME checkable

Proof Idea: automata construction

Corollary: Checking whether a given automatic system exhibits a rotation symmetry is PTIME checkable

Full Symmetry Pattern

All permutations on {1,…,n}

This corresponds to n! automorphisms

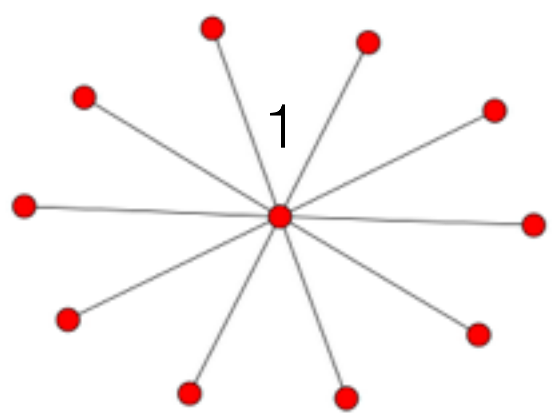
Key: the set of automorphisms forms a group under functional composition generated by:

(1,2) —— a swap

(1,…,n) ——— a rotation

Swap is also regular!

Full Symmetry in a Subsystem



All permutations on {1,…,n} that fix 1

This corresponds to (n-1)! automorphisms

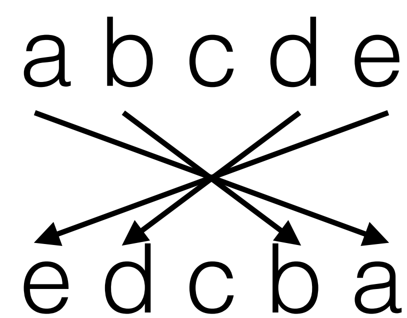
These can be generated by (2,3) and (2,3,….,n)

Verifying full symmetry

Corollary: Checking whether a given automatic system exhibits a full symmetry pattern (in a fixed subsystem) is PTIME checkable

What about reflection?

Unfortunately, it is NOT regular!



You have to compare the first half of the string with the second half of the string

Verifying reflection symmetry

Theorem: Checking whether a given automatic system exhibits a given reflection symmetry pattern is PTIME checkable

Proof idea: introduce a subclass of pushdown automata called

Height-Unambiguous Pushdown Automata

Key Property: they can be synchronised (unlike general PDA)

Automatic symmetry verification extends to huCF patterns

Symmetry Pattern Synthesis

Synthesise-Verify Loop

Synthesise (SAT-solver)

1. Encode Transducers $\mathcal{T}$ as Boolean Formulas
2. Maintain a set $\mathcal{S}$ of boolean constraints that $\mathcal{T}$ has to satisfy
3. Initialise $\mathcal{S}$ to constraints like $\mathcal{T}$ is not trivial, $\mathcal{T}$ is infinite, …

$\mathcal{T}$ →

← NO

C-EX.

Verify (automata method)

1. Is $\mathcal{T}$ a (partial) function?
2. Is $\mathcal{T}$ total?
3. Is $\mathcal{T}$ injective?
4. Is $\mathcal{T}$ surjective?
5. Is $\mathcal{T}$ a homomorphism?

YES ↓

FINISH

"Smart" enumeration of regular symmetry patterns: guess a transducer with 1 state, 2 states, 3 states, 4 states, …

Counterexamples

Three forms of counterexamples:

1. v has to be included in the domain of T
2. w has to be included in the range of T
3. One of two contradictory pairs $T(v,w)$ and $T(v',w')$ must be eliminated.

Each can be encoded as a boolean constraint!

Synthesis of Finite Existential Abstractions (for Proving Safety)

Verify (automata method)

1. Is $\mathcal{T}$ a (partial) function?
2. Is $\mathcal{T}$ total?
3. Is $\mathcal{T}$ injective?
4. Is $\mathcal{T}$ surjective?
5. Is $\mathcal{T}$ a homomorphism?

Relax (3) and (4) in our synthesis-verify loop

Add to Synthesis (boolean constraint):
- "The range of $\mathcal{T}$ finite?"

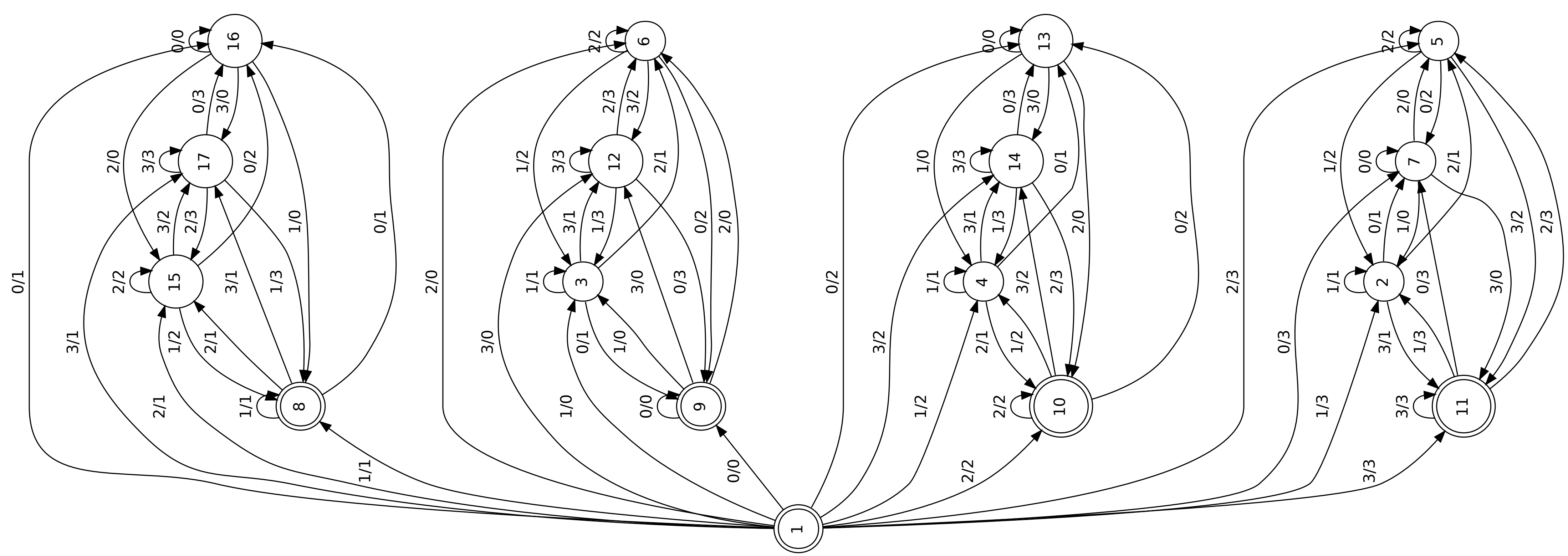
Add to Verify:
- "Does the abstraction satisfy safety?"

Can automatically check safety with a simple fixpoint computation (will terminate since range of $\mathcal{T}$ is finite)

Experiments and Examples

Symmetry Systems (#letters)	# Transducer states	Verif. time	Synth. time
Herman Protocol (2)	5	0.0s	4s
Israeli-Jalfon Protocol (2)	5	0.0s	5s
Gries's Coffee Can (4)	8	0.1s	3m19s
Resource Allocator (3)	11	0.0s	4m56s
Dining Philosopher (4)	17	0.4s	26m

Synthesised Transducer for Dining Philosopher



Conclusion and Future Work

Conclusion

- Look for symmetry patterns instead of symmetries (for an individual instance)
- Expressive symbolic framework for automatically verifying and synthesising symmetry patterns

Future Work

- Synthesis of huCF symmetry patterns
- Synthesis of multiple symmetry patterns